Optimizing Performance for Portable Generic Finite Element Interfaces

Jeremy L Thompson

University of Colorado Boulder

jeremy.thompson@colorado.edu

February 27, 2019

libCEED Team

Developers: Jed Brown[1], Jeremy Thompson[1]
Valeria Barra[1], Tzanio Kolev[2], Jean-Sylvain Camier[2],
Veselin Dobrev[2], Yohann Doudouit[2], Tim Warburton[3],
David Medina[4], & Thilina Rathnayake[5]

Grant: Exascale Computing Project (17-SC-20-SC)

1: University of Colorado, Boulder
2: Lawrence Livermore National Laboratory
3: Virginia Polytechnic Institute and State University
4: OCCA
5: University of Illinois, Urbana-Champaign

Overview

A global sparse matrix is no longer a good representation of high-order linear operators

libCEED is an extensible library that provides a portable algebraic interface and optimized implementations

We have optimized implementations using SIMD intrinsics and LIBXSMM

We have results comparing performance on CEED benchmark problems

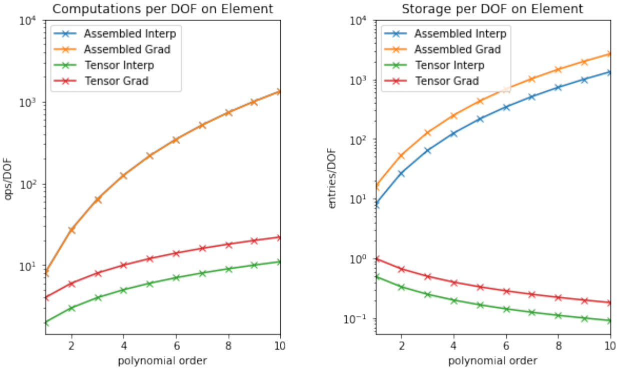
Overview

Center for Efficient Exascale Discretizations

DoE exascale co-design center

- Design discretization algorithms for exascale hardware that deliver significant performance gain over low order methods
- Collaborate with hardware vendors and software projects for exascale hardware and software stack
- Provide efficient and user-friendly unstructured PDE discretization component for exascale software ecosystem

Design Philosophy



Using an assembled matrix inhibits performance optimizations for hexahedral elements

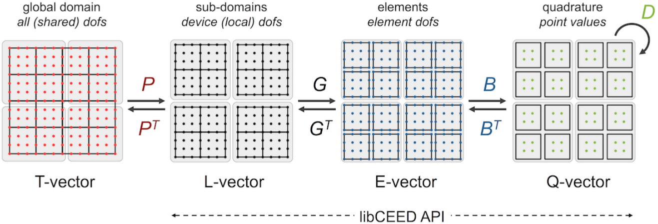
Matrix Free

libCEED design approach:

- Avoid global matrix assembly
- Map each element to reference element
- Geometry data computed on the fly or precomputed
- Easy to parallelize across hetrogeneous nodes

libCEED

$$A = P^T G^T B^T D B G P$$



$$A_L = G^T B^T D B G$$

- G - CeedElemRestriction, local gather/scatter
- B - CeedBasis, provides basis operations such as interp and grad
- D - CeedQFunction, representation of PDE at quadrature points
- A_L - CeedOperator, aggregation of Ceed objects for local action of operator

libCEED

libCEED provides multiple backend implementations

- CPU
  - Pure C
  - Advanced Vector Instructions
  - LIBXSMM
- GPU
  - Pure CUDA
  - OCCA
  - MAGMA

SIMD Intrinsics

- Provides parallelism on multicore CPUs
- Better compiled code without writing assembly
- We target architectures with AVX support

LIBXSMM

- Small matrix multiplication kernels
- $(MNK)^{1/3} \leq 128$
- Targets Intel architectures
- JIT code specialization for compiler independent performance

Vectorization

- Internal vectorization
  - Serial element processing
  - Matrix dimensions may not fit vector length evenly
- External vectorization
  - Blocked element processing
  - Fewer edge cases
  - Gather/scatter needs to interlace elements

Benchmark Problems

Benchmark Problem 1:

- $Bu = f$
- L^2 projection problem

Benchmark Problem 3:

- $Au = f$
- Poisson problem

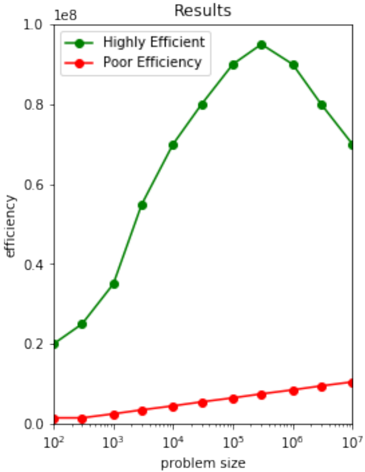
3D scalar problem
$p \in \{1 ... 16\}$, $q = p + 2$
p - polynomial degree, q - number of quadrature nodes
Unpreconditioned CG, maximum of 20 iterations

Compare performance across multiple implementations

Machine Specs

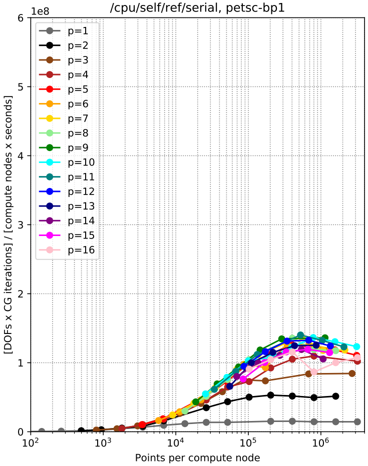
- CU Summit
- Intel Xeon E5-2680 v3, 24 cores/node
- Omni-Path HF1 interconnect
- Using 4 full nodes, no special location

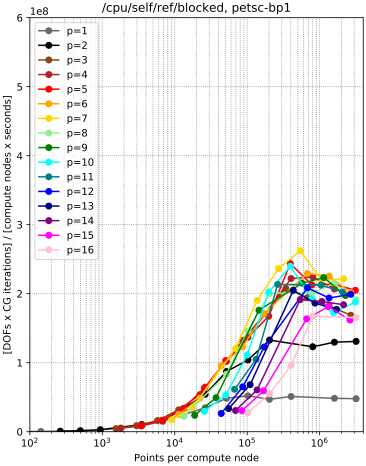
Decoding Results



- Horizontal axis - Problem size, points per compute node
- Vertical axis - Efficiency, [DOFs x CG Iterations] / [Compute Nodes x Seconds]

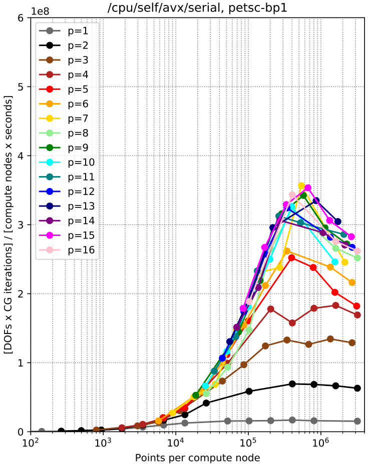
BP 1 Baseline

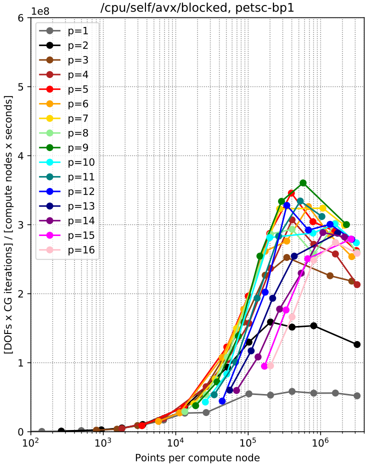




- Better blocked performance

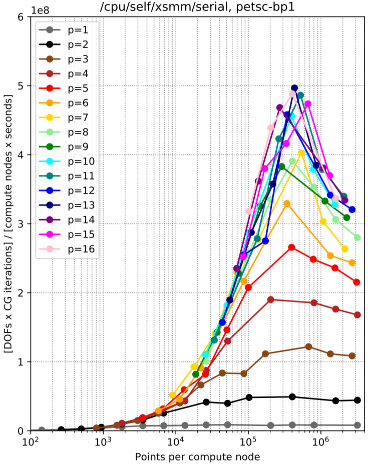
BP 1 AVX Results

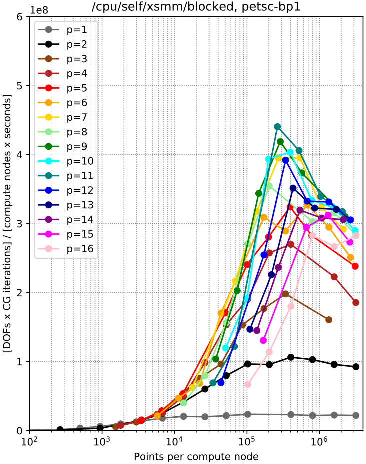




- $p \leq 7$ better blocked performance, $p > 7$ better serial performance

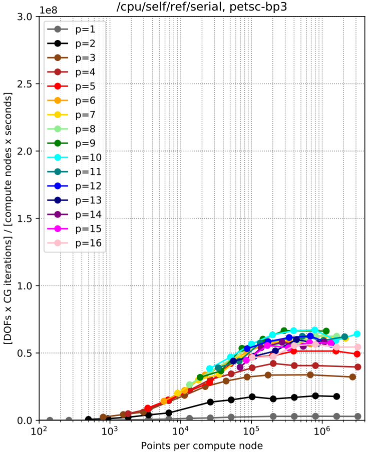
BP 1 LIBXSMM Results

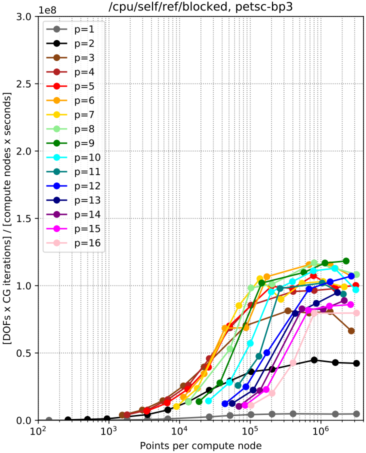




- $p \leq 7$ better blocked performance, $p > 7$ better serial performance
- LIBXSMM handles internal vectorization much better

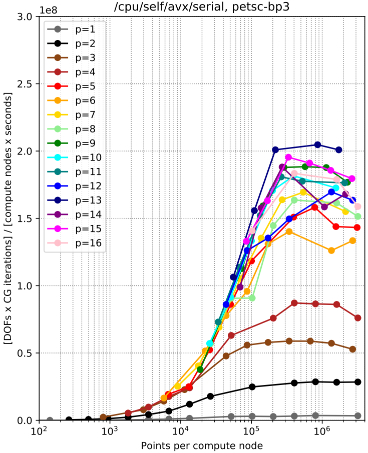
BP 3 Baseline

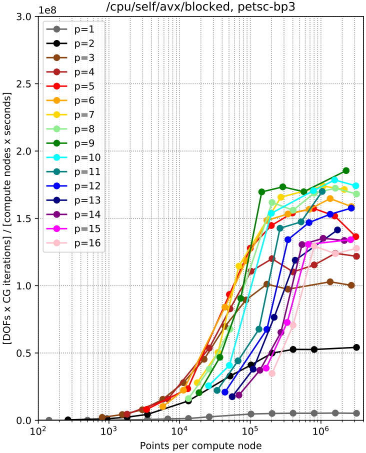




- Better blocked performance

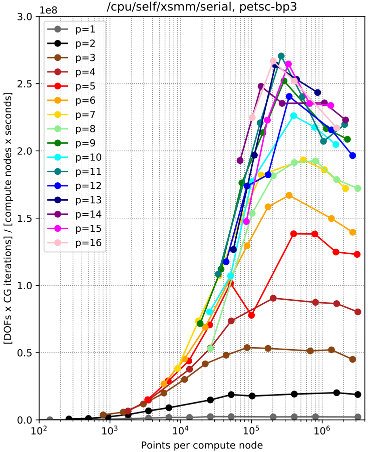
BP 3 AVX Results

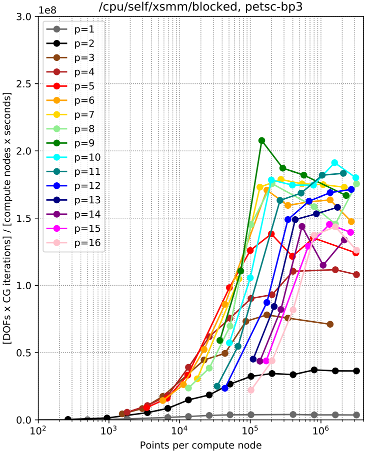




- $p \leq 7$ better blocked performance, $p > 7$ better serial performance

BP 3 LIBXSMM Results





- $p \leq 7$ better blocked performance, $p > 7$ better serial performance
- LIBXSMM handles internal vectorization much better

Future Work

- Further performance tuning
- Improved non-conforming and mixed mesh support
- Algorithmic differentiation of quadrature functions
- We invite contributors and friendly users

Questions?

Advisor: Jed Brown[1]

Collaborators: Valeria Barra[1], Tzanio Kolev[2], Jean-Sylvain Camier[2], Veselin Dobrev[2], Yohann Doudouit[2], Tim Warburton[3], David Medina[4], & Thilina Rathnayake[5]

Grant: Exascale Computing Project (17-SC-20-SC)

1: University of Colorado, Boulder
2: Lawrence Livermore National Laboratory
3: Virginia Polytechnic Institute and State University
4: OCCA
5: University of Illinois, Urbana-Champaign

Optimizing Performance for Portable Generic Finite Element Interfaces

Jeremy L Thompson

University of Colorado Boulder

jeremy.thompson@colorado.edu

February 27, 2019